

SVM learning

WM&R a.a. 2020/21

R. BASILI

DIPARTIMENTO DI INGEGNERIA DELL'IMPRESA

UNIVERSITÀ DI ROMA "TOR VERGATA"

EMAIL: BASILI@INFO.UNIROMA2.IT

Summary

Perceptron Learning

Limitations of linear classifiers

Support Vector Machines

- Maximal margin classification
- Optimization with *hard* margin
- Optimization with *soft* margin

The roles of kernels in SVM-based learning

Linear Classifiers (1)

An hyperplane has equation :

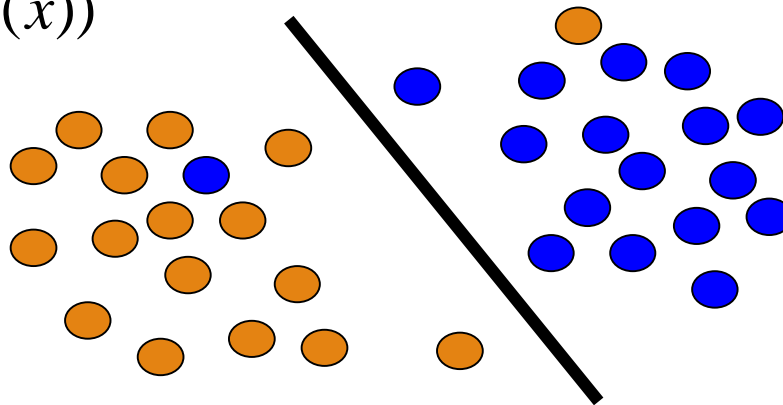
$$f(\vec{x}) = \vec{x} \cdot \vec{w} + b, \quad \vec{x}, \vec{w} \in \mathbb{R}^n, b \in \mathbb{R}$$

\vec{x} is the vector of the instance to be classified

\vec{w} is the hyperplane gradient

Classification function:

$$h(x) = \text{sign}(f(x))$$



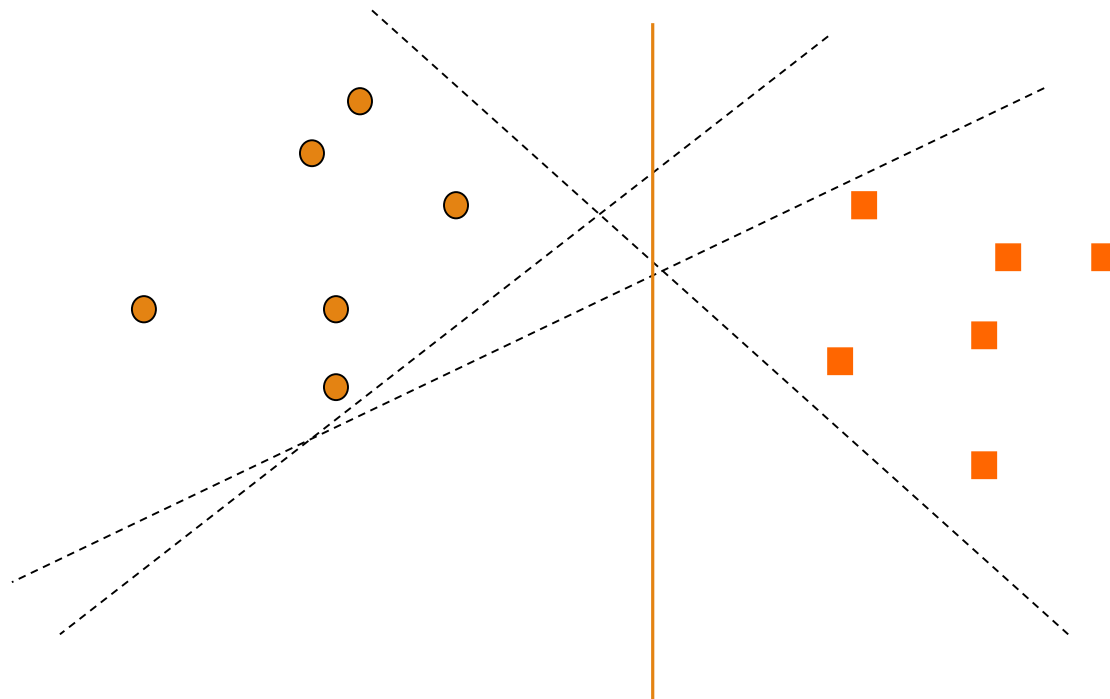
Linear Classifiers (2)

Computationally simple.

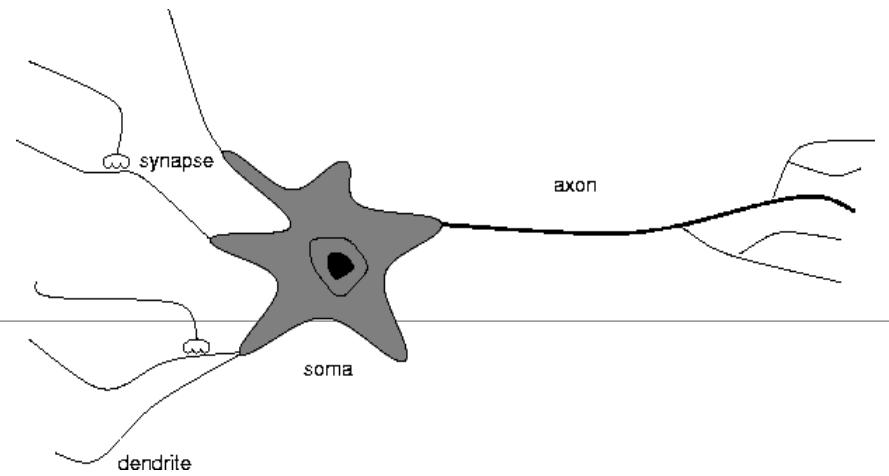
Basic idea: select an hypothesis that makes no mistake over training-set.

The separating function is equivalent to a neural net with just one neuron (perceptron)

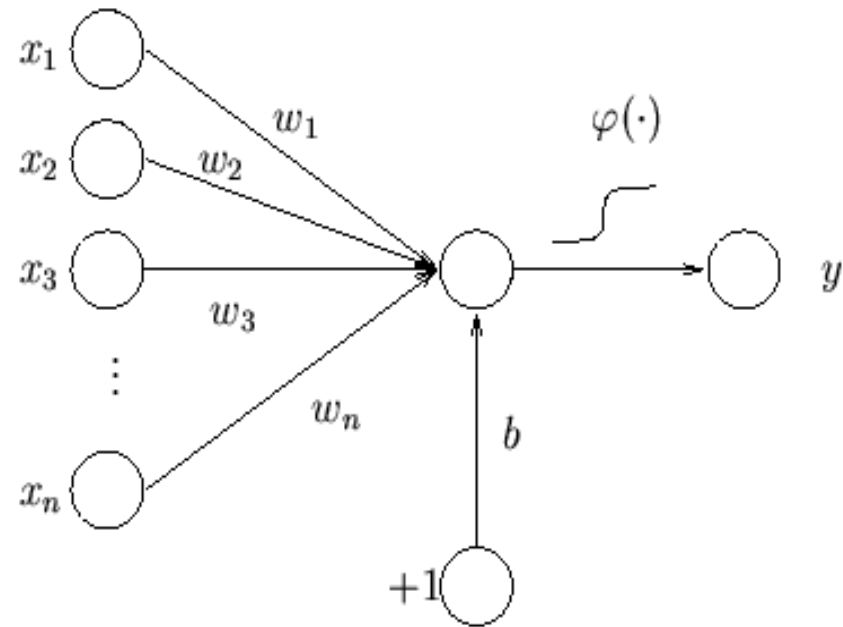
Which hyperplane?



Perceptron



$$\varphi(\vec{x}) = \text{sgn}\left(\sum_{i=1..n} w_i \times x_i + b\right)$$



Notation

The functional margin of an example (\vec{x}_i, y_i)

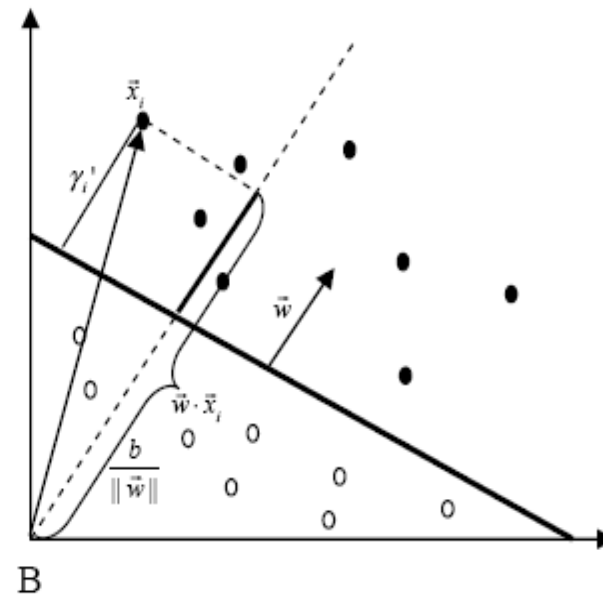
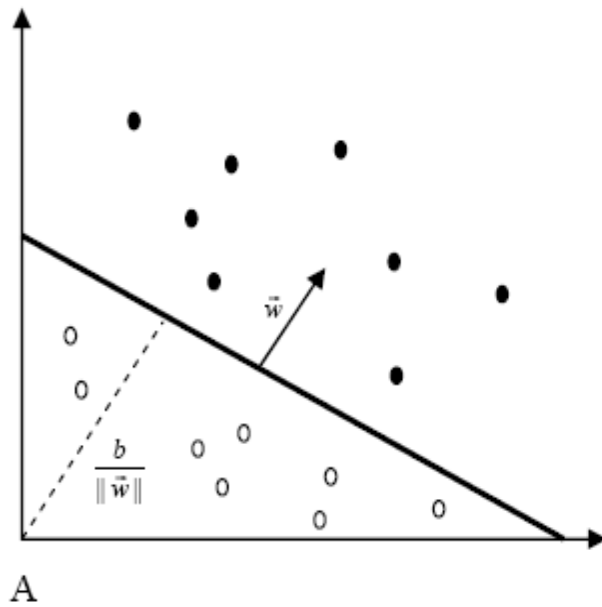
with respect to an hyperplane is:

$$\gamma_i = y_i(\vec{w} \cdot \vec{x}_i + b)$$

The distribution of functional margins of an hyperplane (\vec{w}, b) with respect to a training set S is the distribution of margins of the examples in S .

The functional margin of an hyperplane (\vec{w}, b) with respect to S is the minimum margin of the distribution

Geometric Margin



Inner product and cosine distance

From

$$\cos(\vec{x}, \vec{w}) = \frac{\vec{x} \cdot \vec{w}}{\|\vec{x}\| \cdot \|\vec{w}\|}$$

It follows that:

$$\|\vec{x}\| \cos(\vec{x}, \vec{w}) = \frac{\vec{x} \cdot \vec{w}}{\|\vec{w}\|} = \vec{x} \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

Norm of \vec{x} times \vec{x} cosine \vec{w} , i.e. the projection of \vec{x} onto \vec{w}

Notations (2)

By normalizing the hyperplan equation, i.e. $\left(\frac{\vec{w}}{\|\vec{w}\|}, \frac{b}{\|\vec{w}\|} \right)$
we get the geometrical margin

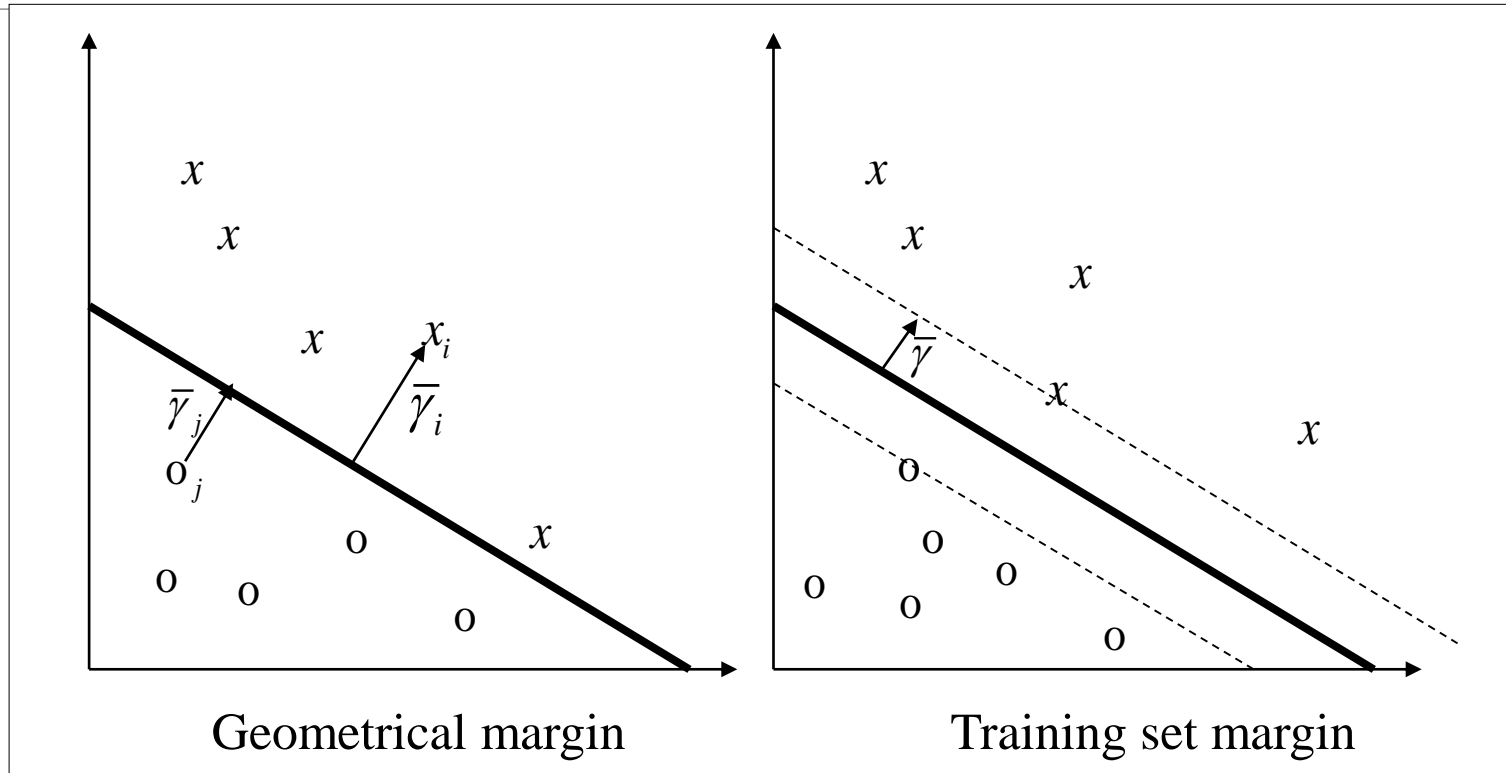
$$\gamma_i = y_i(\vec{w} \cdot \vec{x}_i + b)$$

The geometrical margin corresponds to the distance of points in S from the hyperplane.

For example in \mathbb{R}^2

$$d(P, r) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

Geometric margin vs. data points in the training set

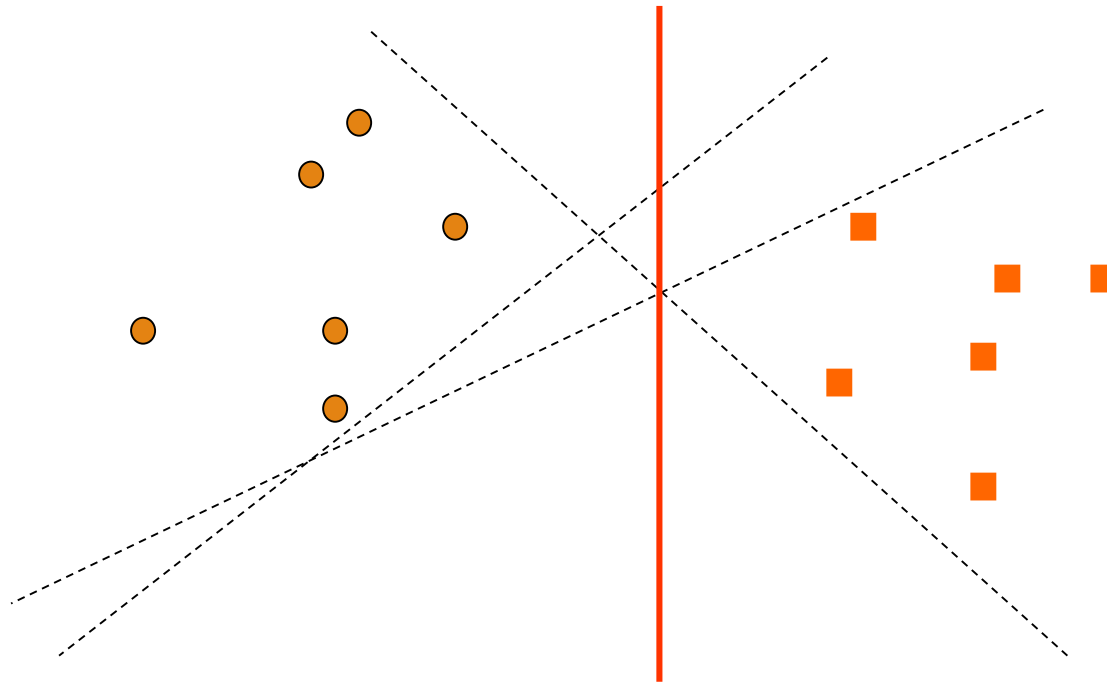


Notations (3)

The margin of the training set S is the maximal geometric margin among every hyperplane.

The hyperplane that corresponds to this (maximal) margin is called *maximal margin hyperplane*

Maximal margin vs other margins



Perceptron: on-line algorithm

$$\vec{w}_0 \leftarrow \vec{0}; b_0 \leftarrow 0; k \leftarrow 0; R \leftarrow \max_{1 \leq i \leq l} \|\vec{x}_i\|$$

Repeat

for $i = 1$ to ℓ

if $y_i (\vec{w}_k \cdot \vec{x}_i + b_k) \leq 0$ then

$$\vec{w}_{k+1} = \vec{w}_k + \eta y_i \vec{x}_i$$

$$b_{k+1} = b_k + \eta y_i R^2$$

$$k = k + 1$$


endif

endfor

until no error is found

return $k, (\vec{w}_k, b_k)$

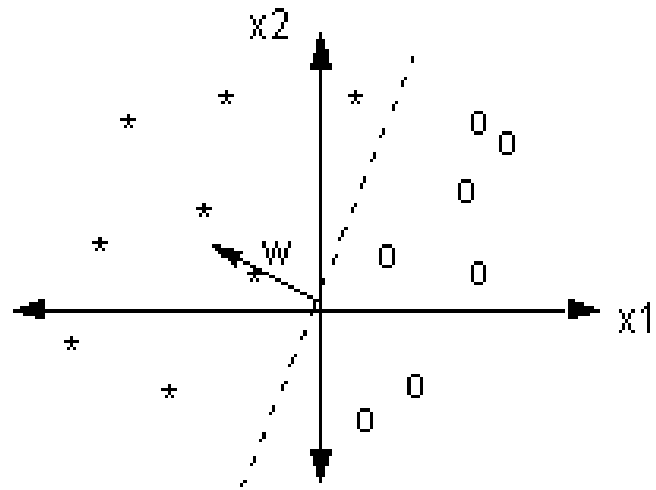
Classification
Error



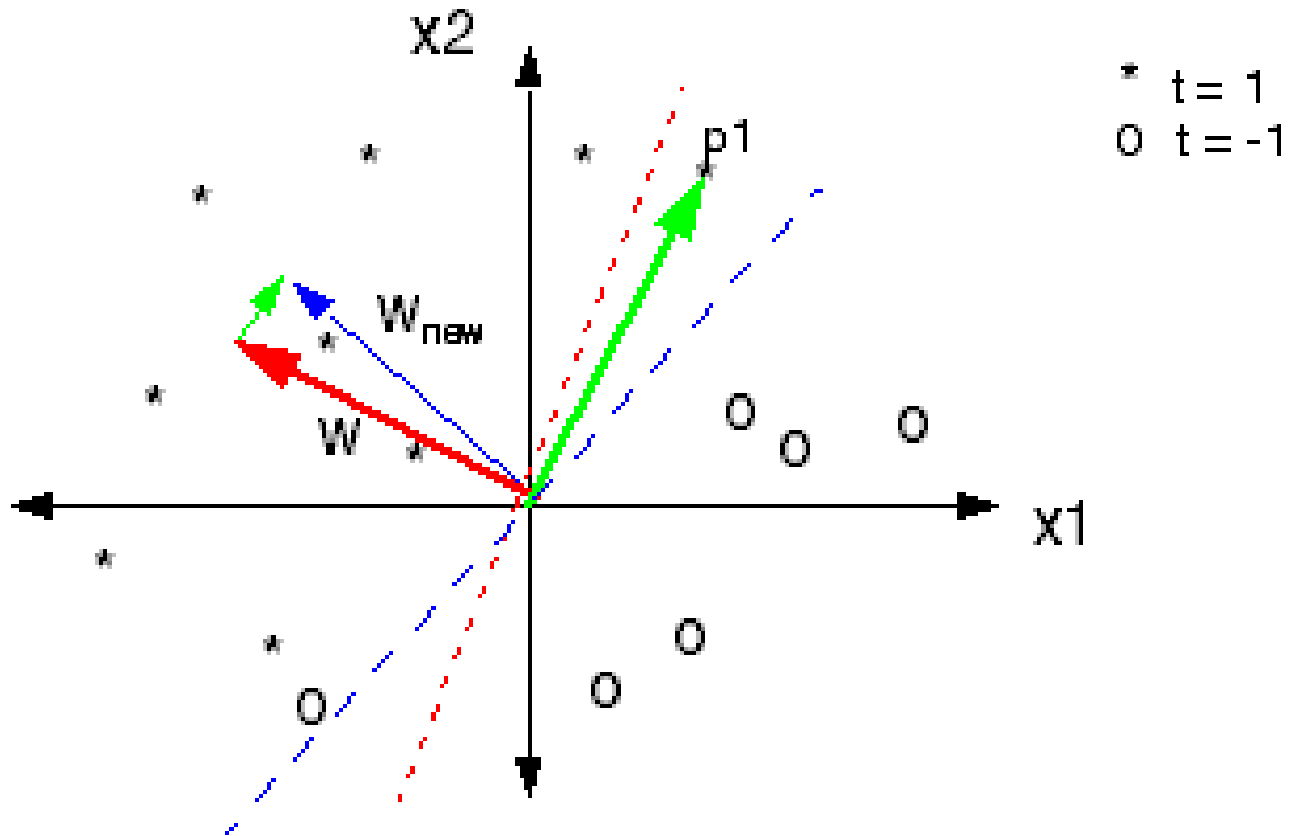
adjustments



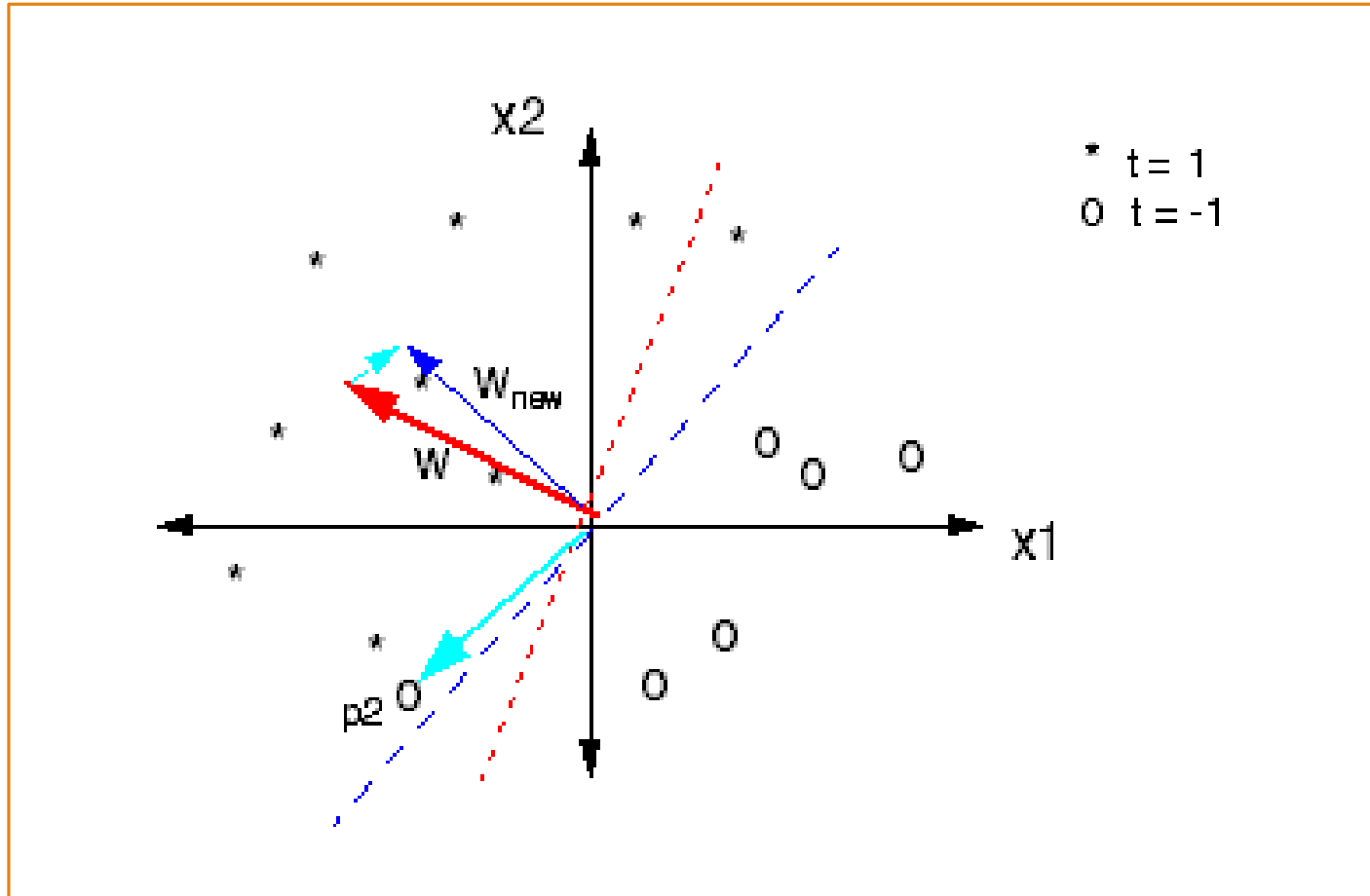
The mechanics of the perceptron



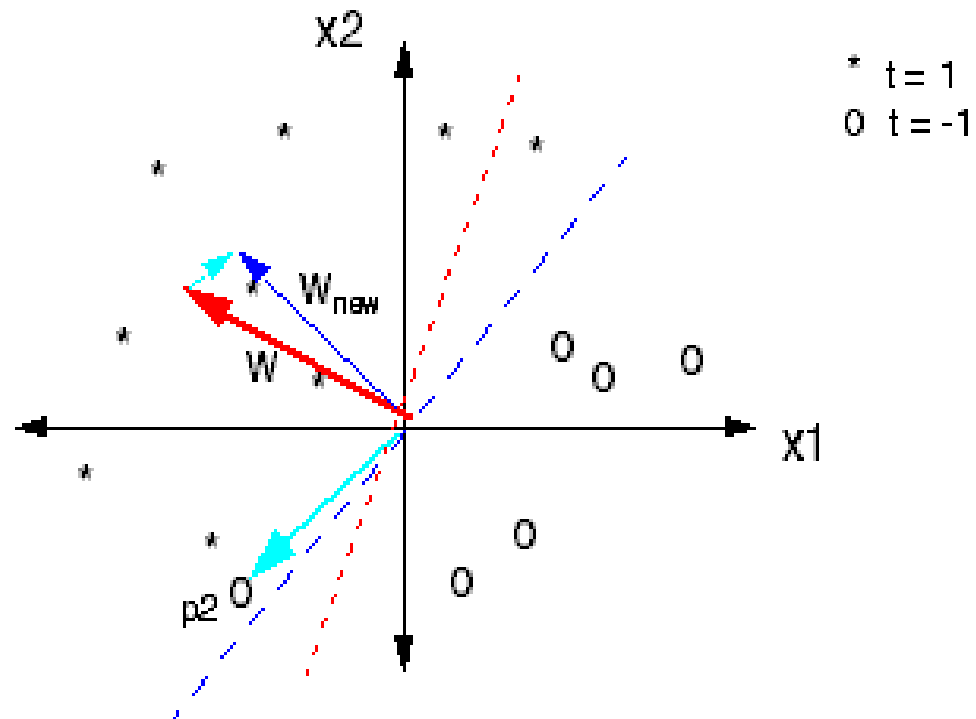
On-line Learning: dealing with positive examples



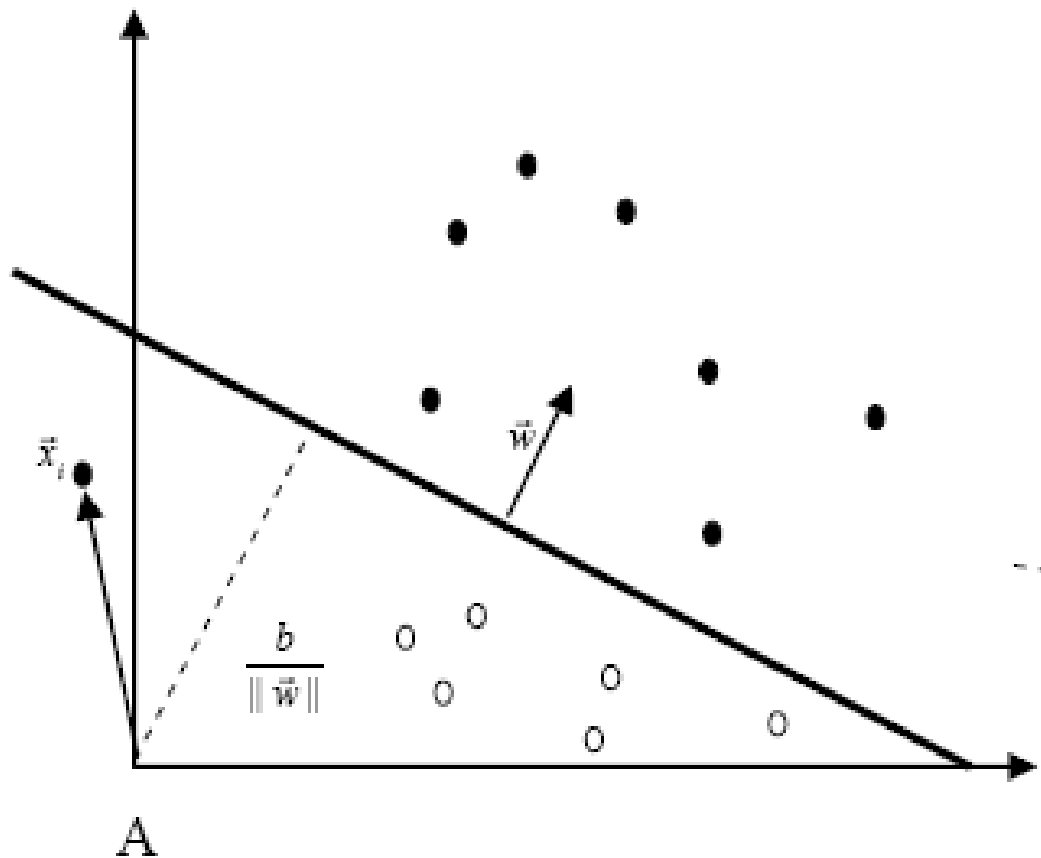
On-line Learning: dealing with negative examples



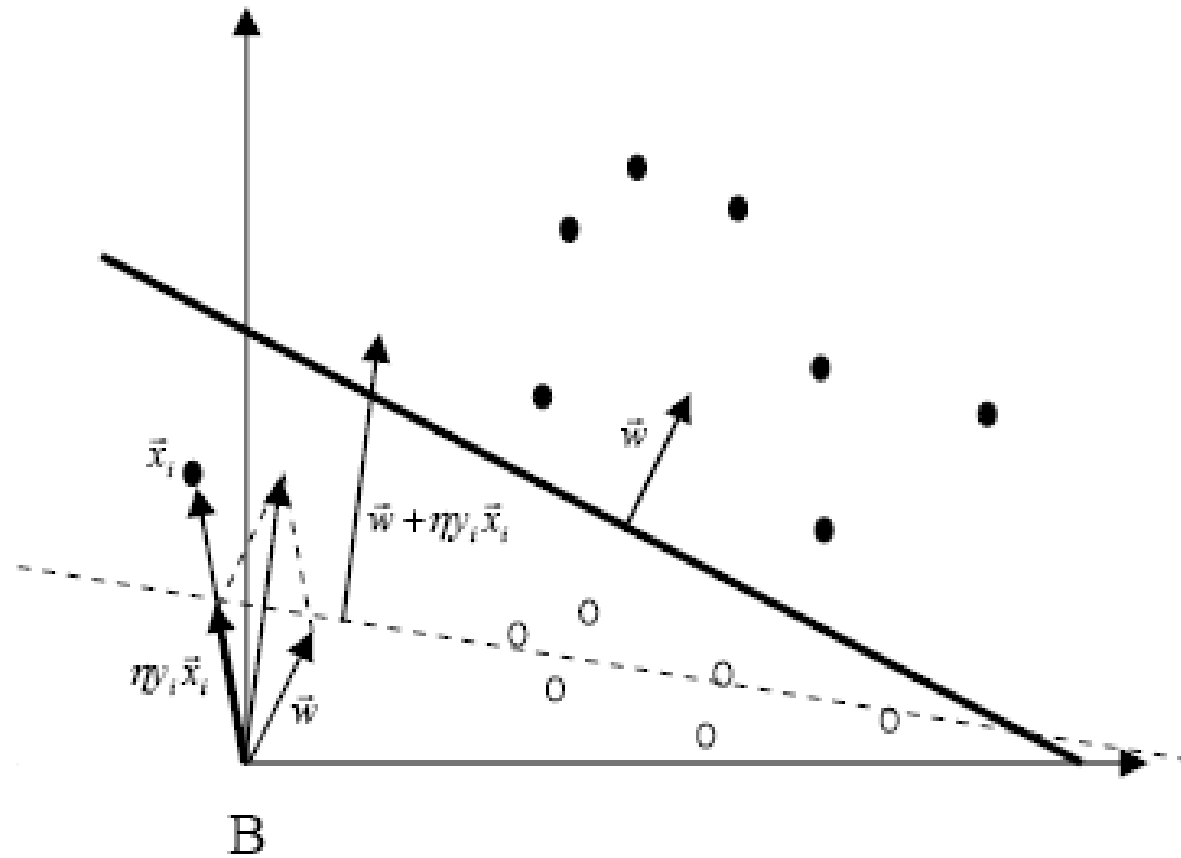
The adjusted hyperplane



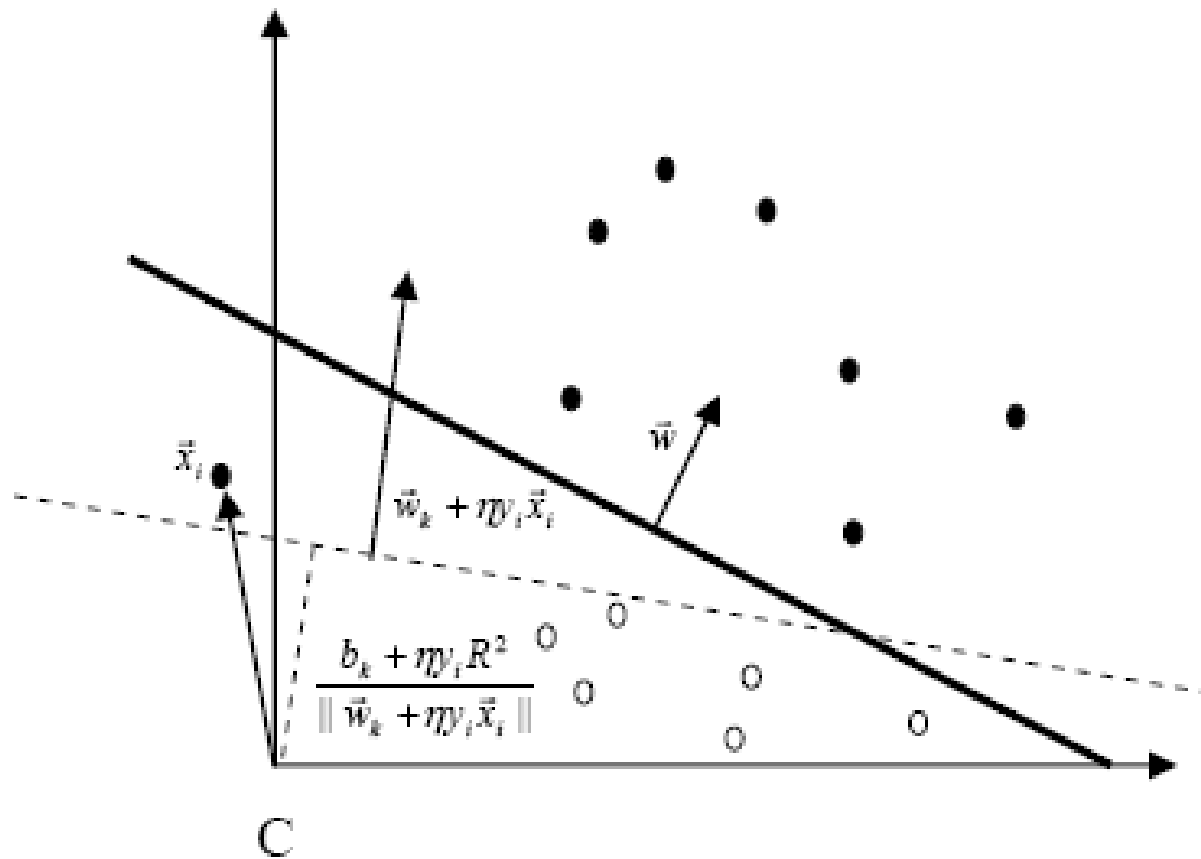
Perceptron: the management of an individual instance x



Adjusting the (hyper)plane directions



Adjusting the distance from the origins



Novikoff theorem

Given a non-trivial training-set S ($|S|=m \gg 0$) and:

$$R = \max_{1 \leq i \leq l} \| \mathbf{x}_i \| .$$

If a vector \mathbf{w}^* , $\| \mathbf{w}^* \| = 1$, exists such that:

$$y_i (\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \geq \gamma \quad i = 1, \dots, m,$$

with $\gamma > 0$, then the maximal number of errors made by the perceptron is :

$$t^* = \left(\frac{2R}{\gamma} \right)^2 ,$$

Consequences

The theorem states that whatever is the length of the geometrical margin, if data instances are **linearly separable**, then the **perceptron** is able to find the separating hyperplane in a **finite number of steps**.

This number is inversely proportional to the square of the margin.

This **bound is invariant** to the scale of individual *patterns*.

The **learning rate** is not critical but only affects the rate of convergence.

Duality

The decision function of linear classifiers can be written as follows:

$$h(x) = \text{sgn}(\vec{w} \cdot \vec{x} + b) = \text{sgn}\left(\sum_{j=1 \dots m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b\right) = \\ \text{sgn}\left(\left(\sum_{i=1 \dots m} \alpha_j y_j \vec{x}_j \cdot \vec{x}\right) + b\right)$$

as well the adjustment function

$$\text{if } y_i \left(\sum_{j=1 \dots m} \alpha_j y_j \vec{x}_j \cdot \vec{x}_i + b\right) \leq 0 \quad \text{then } \alpha_i = \alpha_i + \eta$$

The learning rate η impacts only in the re-scaling of the hyperplanes, and does not influence the algorithm ($\eta = 1$).

\Rightarrow Training data only appear in the scalar products!!

First property of SVMs

DUALITY is the first property of Support Vector Machines

The SVMs are learning machines of the kind:

$$f(x) = \text{sgn}(\vec{w} \cdot \vec{x} + b) = \text{sgn}\left(\sum_{j=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b\right)$$

It must be noted that (input, i.e. training & testing instances) data only appear in the scalar product

The matrix $G = \left(\langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle\right)_{i,j=1}^l$ is called **Gram matrix** of the incoming distribution

Limitations of linear classifiers

- Problems in dealing with non linearly separable data
- Treatment of Noisy Data
- Data must be in real-value vector formalism, i.e. a underlying metric space topology is required

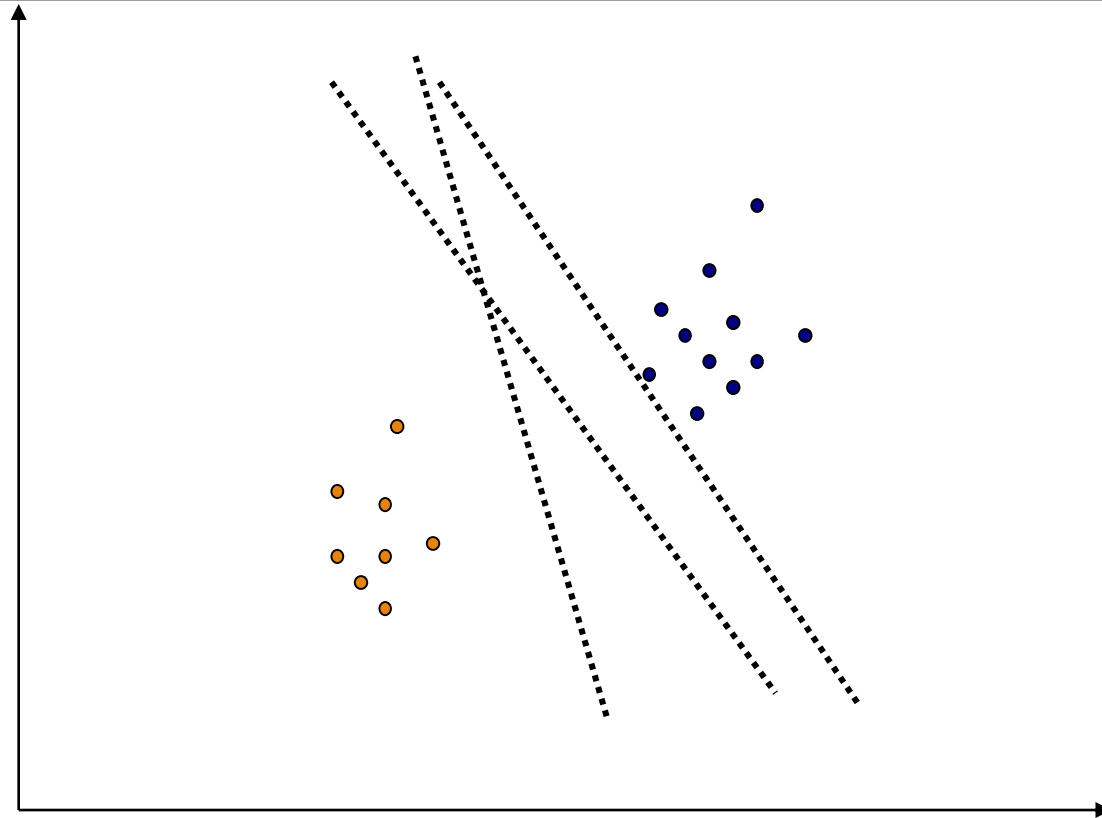
Solutions

Artificial Neural Networks (ANN) approach: augment the number of neurons, and organize them into layers \Rightarrow multilayer neural networks \Rightarrow Learning through the Back-propagation algorithm (Rumelhart & McLelland, 91).

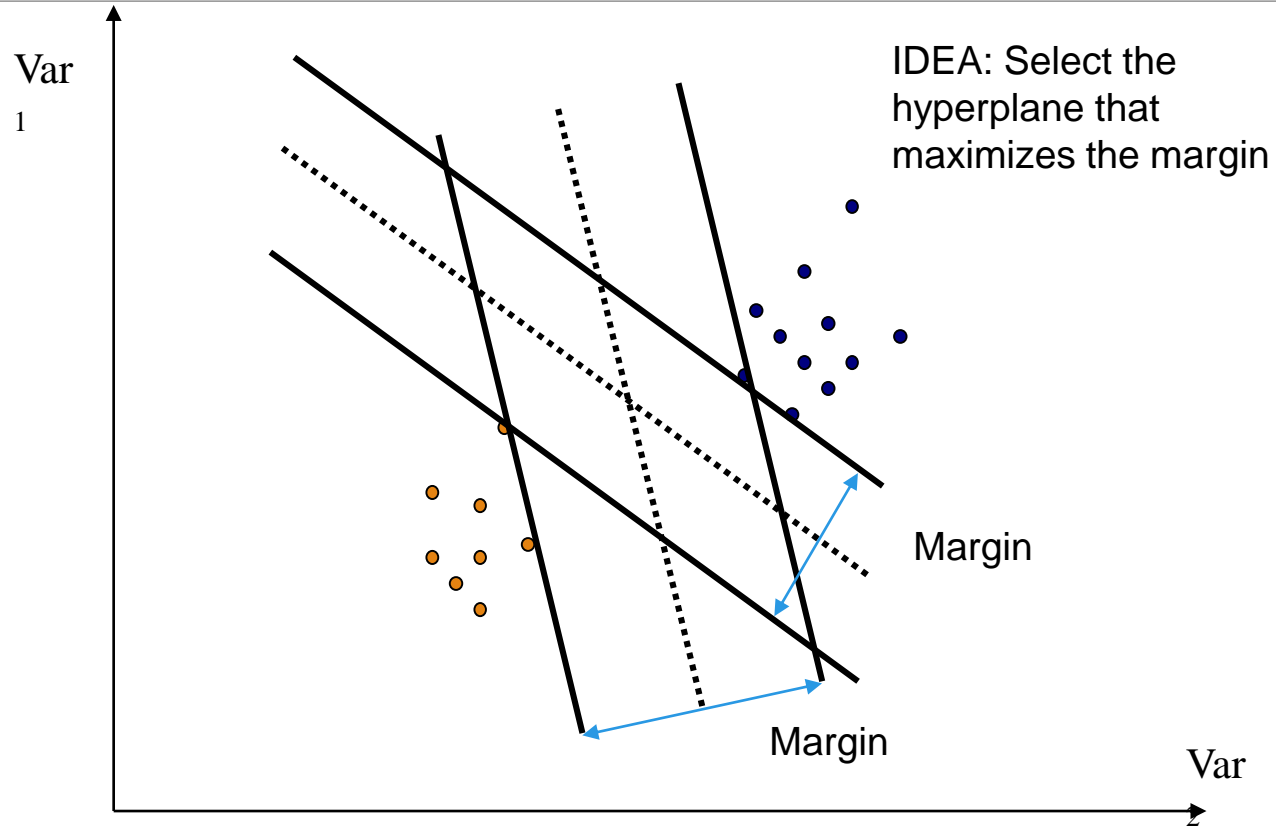
SVMs approach: Extend the representation by exploiting kernel functions (i.e. non linear often task dependent functions described by the Gram matrix).

- In this way the learning algorithms are decoupled from the application domain, that can be coded exclusively through task-specific kernel functions.
- The feature modeling does not necessarily have to produce real-valued vectors but can be derived from intrinsic properties of the training objects
- Complex data structures, e.g. sequences, trees, graphs or PCA-like decompositions (e.g. LSA), can be managed by individual kernels

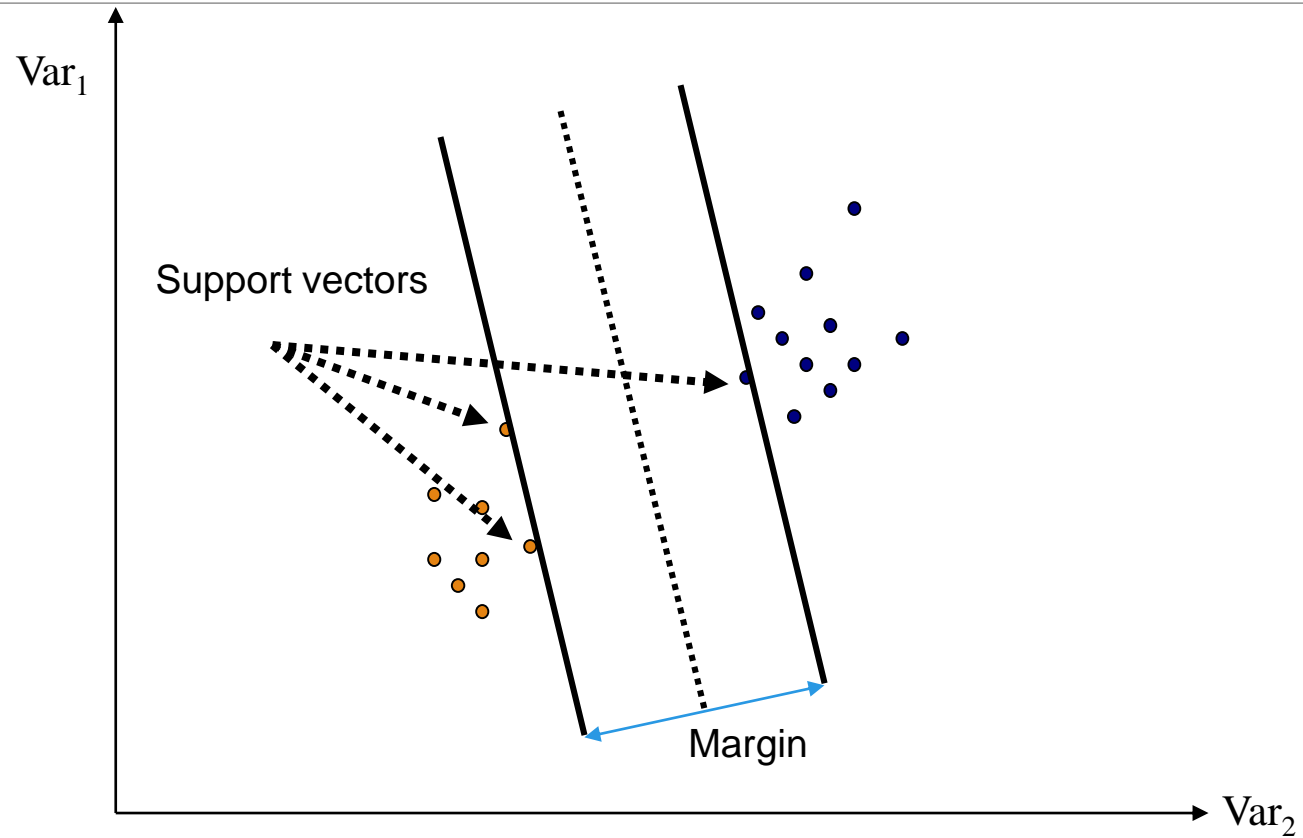
Which hyperplane?



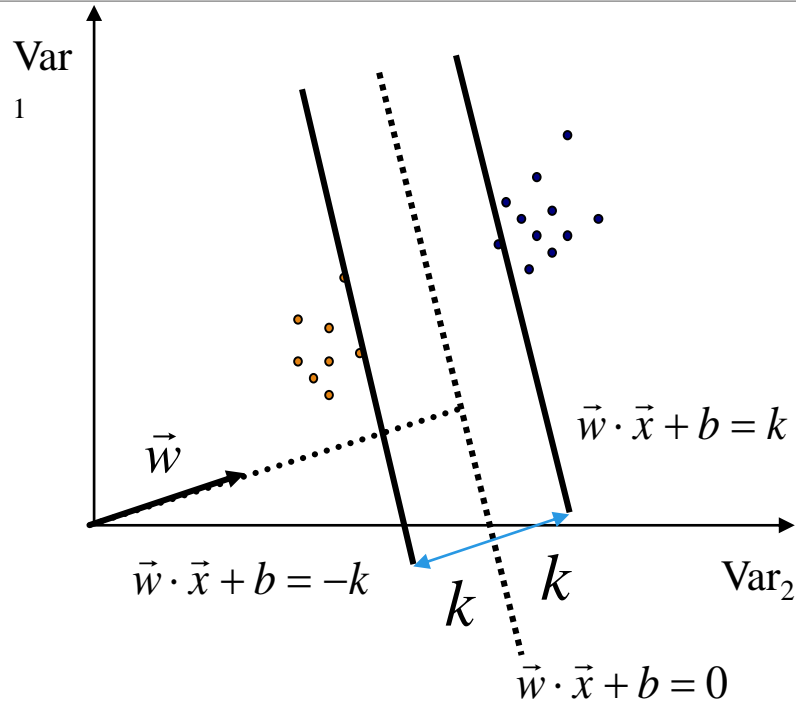
Maximum Margin Hyperplanes



Support Vectors



How to get the maximum margin?



The geometric margin
is:

$$\frac{2|k|}{\|\vec{w}\|}$$

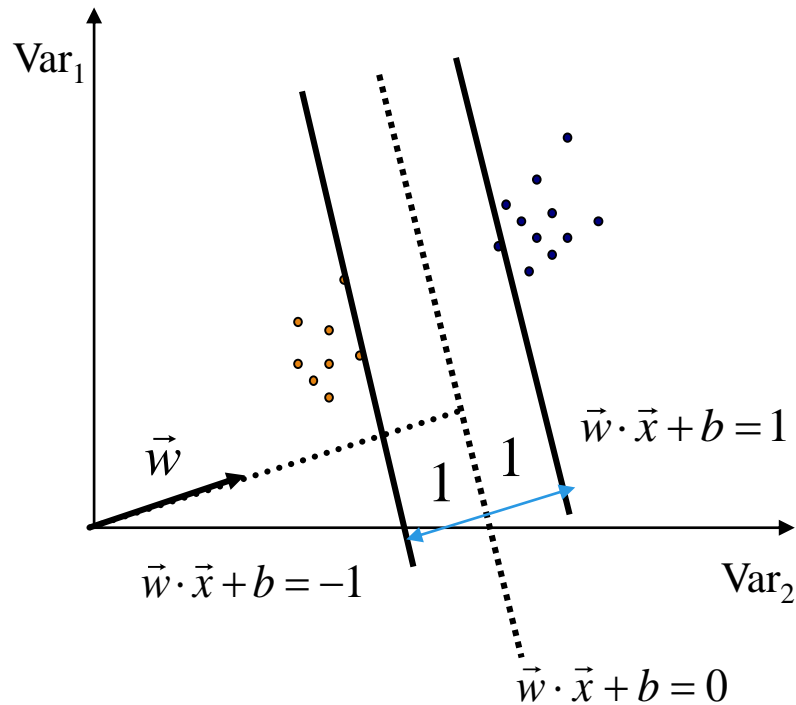
Optimization problem

$$\text{MAX} \frac{2|k|}{\|\vec{w}\|}$$

$\vec{w} \cdot \vec{x} + b \geq +k$, if \vec{x} is a positive ex.

$\vec{w} \cdot \vec{x} + b \leq -k$, se \vec{x} is a negativ ex.

Scaling the hyperplane ...



There is a scale for which $k=1$.

The optimization problem becomes:

$$\max \frac{2}{\|\vec{w}\|}$$

$$\vec{w} \cdot \vec{x} + b \geq +1, \text{ if } \vec{x} \text{ is positive}$$

$$\vec{w} \cdot \vec{x} + b \leq -1, \text{ if } \vec{x} \text{ is negative}$$

The optimization problem

The optimal hyperplane satisfies:

- Minimize $\tau(\vec{w}) = \frac{1}{2} \|\vec{w}\|^2$
- Under: $y_i ((\vec{w} \cdot \vec{x}_i) + b) \geq 1 \quad i = 1, \dots, m$

The dual problem is simpler

Definition of the Lagrangian

Def. 2.24 Let $f(\vec{w})$, $h_i(\vec{w})$ and $g_i(\vec{w})$ be the objective function, the equality constraints and the inequality constraints (i.e. \geq) of an optimization problem, and let $L(\vec{w}, \vec{\alpha}, \vec{\beta})$ be its Lagrangian, defined as follows:

$$L(\vec{w}, \vec{\alpha}, \vec{\beta}) = f(\vec{w}) - \sum_{i=1}^m \alpha_i g_i(\vec{w}) - \sum_{i=1}^l \beta_i h_i(\vec{w})$$

$$f(\vec{w}) = \tau(\vec{w}) = \frac{1}{2} \|\vec{w}\|^2$$

$$y_i ((\vec{w} \cdot \vec{x}_i) + b) \geq 1, \quad i = 1, \dots, l$$

$\vec{\beta}$ are not used as no equality constraint is needed in the primal equation

Dual optimization problem

The Lagrangian dual problem of the above primal problem is

$$\text{maximize } \theta(\vec{\alpha}, \vec{\beta})$$

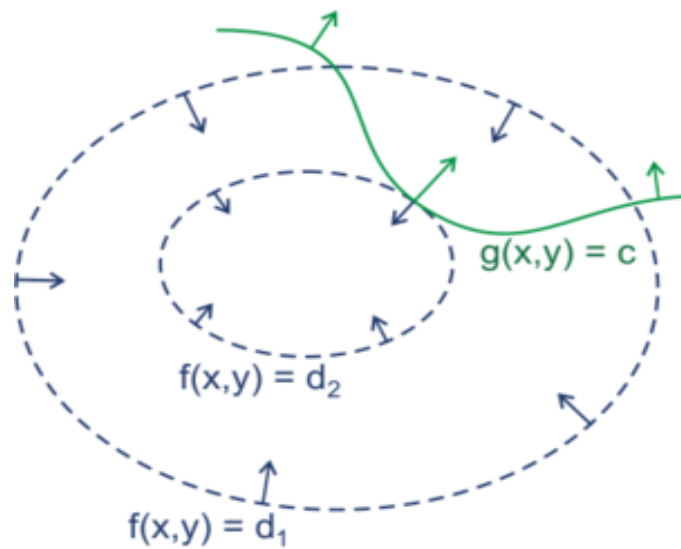
$$\text{subject to } \vec{\alpha} \geq \vec{0}$$

$$\text{where } \theta(\vec{\alpha}, \vec{\beta}) = \inf_{w \in W} L(\vec{w}, \vec{\alpha}, \vec{\beta})$$

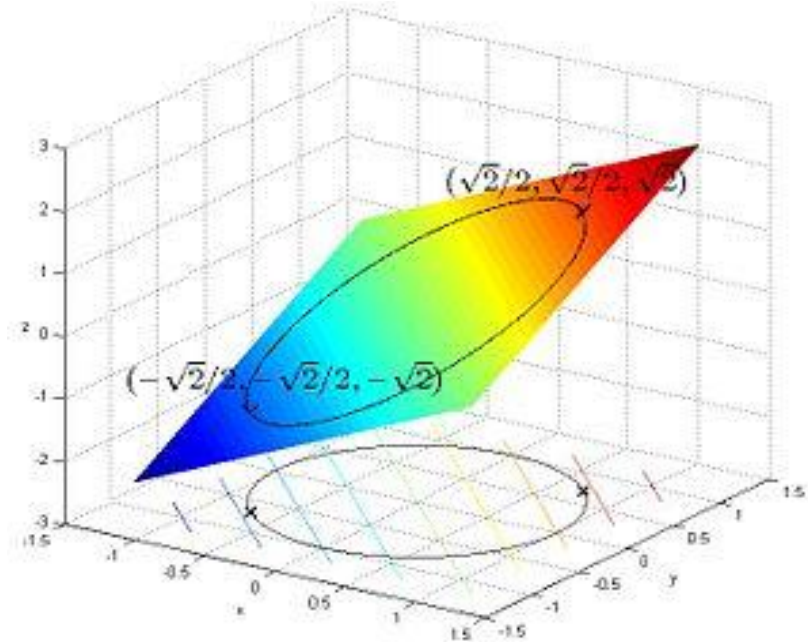
Notice that the multipliers $\vec{\beta}$ are not used in the dual optimization problem as no equality constraint is imposed in the primal form

Graphically:

Two examples of constrained optimization (with equalities)



$$f(x, y) = x^2 + y^2$$
$$g(x, y) = c$$



$$f(x, y) = x + y$$
$$g(x, y) = x^2 + y^2 - 1$$

Dual Optimization problem

The Lagrangian dual problem of the above primal problem is

$$\text{maximize } \theta(\vec{\alpha}, \vec{\beta})$$

$$\text{subject to } \vec{\alpha} \geq \vec{0}$$

$$\text{where } \theta(\vec{\alpha}, \vec{\beta}) = \inf_{w \in W} L(\vec{w}, \vec{\alpha}, \vec{\beta})$$

Notice that the multipliers $\vec{\beta}$ are not used in the targeted optimization as no equality constraint is imposed

Transforming into the dual

The Lagrangian corresponding to our problem becomes:

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} \vec{w} \cdot \vec{w} - \sum_{i=1}^m \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1]$$

In order to solve the dual problem we compute

$$\theta(\vec{\alpha}, \vec{\beta}) = \inf_{w \in W} L(\vec{w}, \vec{\alpha}, \vec{\beta})$$

and then imposing derivatives to 0, wrt \vec{w}

Transforming into the dual (cont.)

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} \vec{w} \cdot \vec{w} - \sum_{i=1}^m \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1]$$

Imposing derivatives = 0 wrt \vec{w}

$$\frac{\partial L(\vec{w}, b, \vec{\alpha})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^m y_i \alpha_i \vec{x}_i = \vec{0} \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^m y_i \alpha_i \vec{x}_i$$

and wrt b

$$\frac{\partial L(\vec{w}, b, \vec{\alpha})}{\partial b} = \sum_{i=1}^m y_i \alpha_i = 0$$

Transforming into the dual (cont.)

$$\vec{w} = \sum_{i=1}^m y_i \alpha_i \vec{x}_i \quad \frac{\partial L(\vec{w}, b, \vec{\alpha})}{\partial b} = \sum_{i=1}^m y_i \alpha_i = 0$$

... by substituting into the objective function

$$\begin{aligned} L(\vec{w}, b, \vec{\alpha}) &= \frac{1}{2} \vec{w} \cdot \vec{w} - \sum_{i=1}^m \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1] = \\ &= \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j - \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j + \sum_{i=1}^m \alpha_i \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j \end{aligned}$$

Dual Optimization problem

$$\text{maximize} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j$$

$$\text{subject to} \quad \alpha_i \geq 0, \quad i = 1, \dots, m$$

$$\sum_{i=1}^m y_i \alpha_i = 0$$

- The formulation depends on the set of variables $\underline{\alpha}$ and not from \underline{w} and b
- It has a simpler form
- It makes explicit the individual contributions (α_i) of (a selected set of) examples (x_i)

Khun-Tucker Theorem

Necessary (and sufficient) conditions for the existence of the optimal solution are the following:

$$\frac{\partial L(\vec{w}^*, \vec{\alpha}^*, \vec{\beta}^*)}{\partial \vec{w}} = \vec{0}$$

$$\frac{\partial L(\vec{w}^*, \vec{\alpha}^*, \vec{\beta}^*)}{\partial \vec{\beta}} = \vec{0}$$

$$\begin{aligned} \alpha_i^* g_i(\vec{w}^*) &= 0, & i = 1, \dots, m \\ g_i(\vec{w}^*) &\leq 0, & i = 1, \dots, m \\ \alpha_i^* &\geq 0, & i = 1, \dots, m \end{aligned}$$

Karush-Kuhn-Tucker constraint

$$\begin{aligned} \vec{w} &= \sum_{i=1}^m y_i \alpha_i \vec{x}_i \\ \sum_{i=1}^m y_i \alpha_i &= 0 \end{aligned}$$

Some consequences

Lagrange constraints: $\sum_{i=1}^m \alpha_i y_i = 0 \quad \vec{w} = \sum_{i=1}^m \alpha_i y_i \vec{x}_i$

Karush-Kuhn-Tucker constraints

$$\alpha_i \cdot [y_i (\vec{x}_i \cdot \vec{w} + b) - 1] = 0, \quad i = 1, \dots, m$$

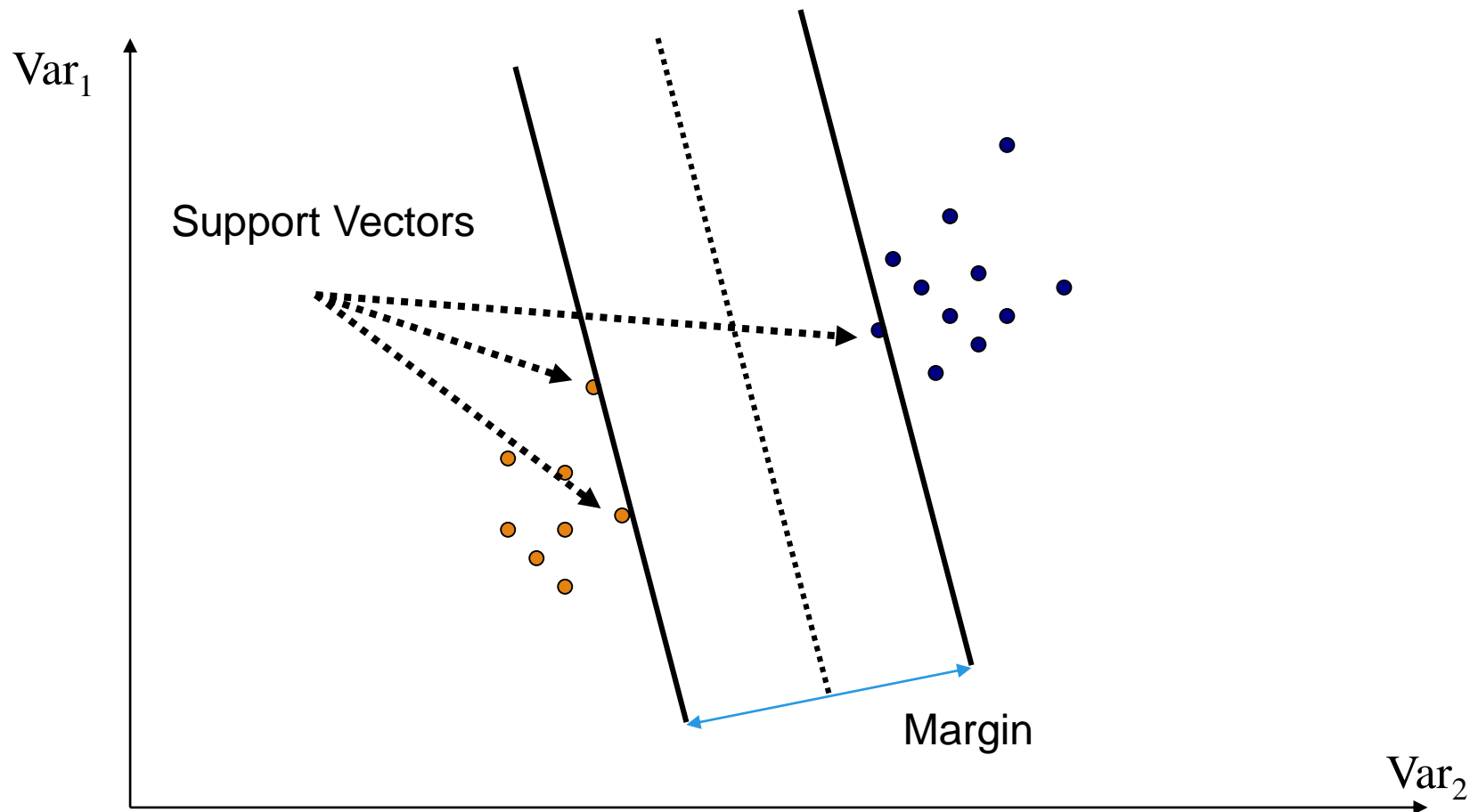
The support vector are \vec{x}_i having not null α_i , i.e. such that $y_i (\vec{x}_i \cdot \vec{w} + b) = -1$

They lie on the **frontier**

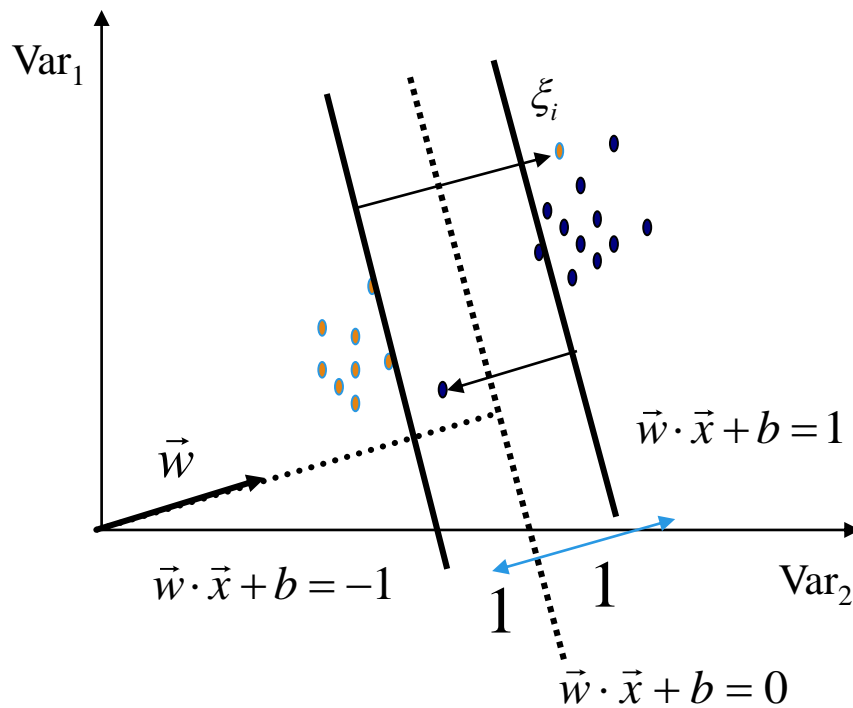
b is derived through the following formula

$$b^* = -\frac{\vec{w}^* \cdot \vec{x}^+ + \vec{w}^* \cdot \vec{x}^-}{2}$$

Support Vectors



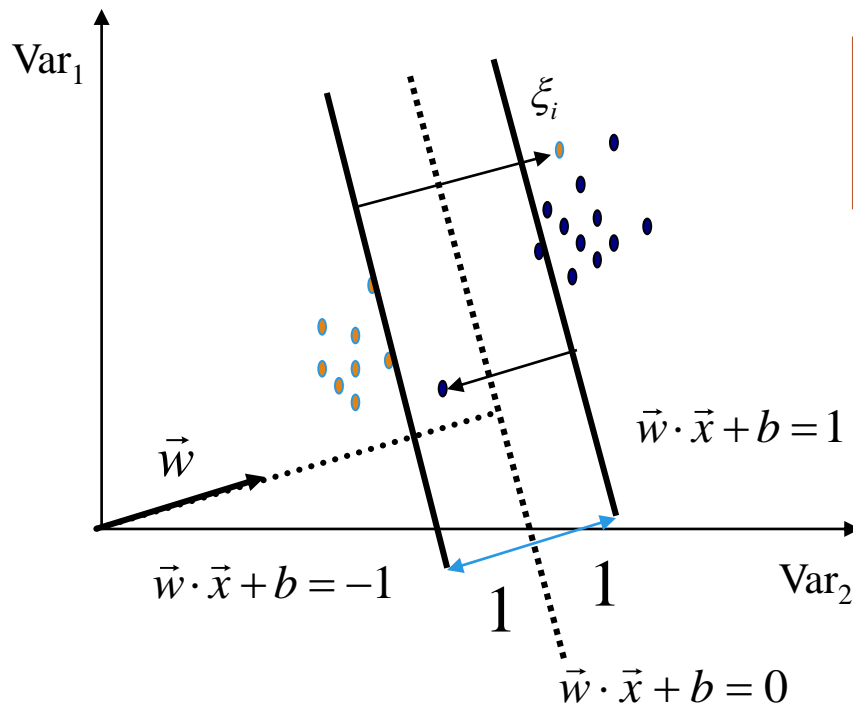
Non linearly separable training data



Slack variables ξ_i are introduced

Mistakes are allowed and the optimization function is penalized

Soft Margin SVMs



New constraints:

$$y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \quad \forall \vec{x}_i$$
$$\xi_i \geq 0$$

Objective function:

$$\min \frac{1}{2} \|\vec{w}\|^2 + C \sum_i \xi_i$$

C is the *trade-off* between margin and errors

Converting in the dual form

$$\begin{cases} \min & \|\vec{w}\| + C \sum_{i=1}^m \xi_i^2 \\ & y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i, \quad \forall i = 1, \dots, m \\ & \xi_i \geq 0, \quad i = 1, \dots, m \end{cases}$$

$$L(\vec{w}, b, \vec{\xi}, \vec{\alpha}) = \frac{1}{2} \vec{w} \cdot \vec{w} + \frac{C}{2} \sum_{i=1}^m \xi_i^2 - \sum_{i=1}^m \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1]$$

deriving wrt $\vec{w}, \vec{\xi}$ and b

Partial derivatives

$$\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^m y_i \alpha_i \vec{x}_i = \vec{0} \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^m y_i \alpha_i \vec{x}_i$$

$$\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial \vec{\xi}} = C \vec{\xi} - \vec{\alpha} = \vec{0}$$

$$\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial b} = \sum_{i=1}^m y_i \alpha_i = 0$$

Substitution in the objective function

$$\begin{aligned} &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j + \frac{1}{2C} \vec{a} \cdot \vec{a} - \frac{1}{C} \vec{a} \cdot \vec{a} = \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j - \frac{1}{2C} \vec{a} \cdot \vec{a} = \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j + \frac{1}{C} \delta_{ij} \right), \end{aligned}$$

δ_{ij} of Kronecker

Dual optimization problem (the final form)

$$\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j + \frac{1}{C} \delta_{ij})$$

$$\alpha_i \geq 0, \quad \forall i = 1, \dots, m$$

$$\sum_{i=1}^m y_i \alpha_i = 0$$

Soft Margin Support Vector Machines

$$\min \frac{1}{2} \|\vec{w}\|^2 + C \sum_i \xi_i \quad \begin{array}{l} y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \quad \forall \vec{x}_i \\ \xi_i \geq 0 \end{array}$$

The algorithm tries to keep $\xi_i = 0$ and then maximizes the margin.

The algorithm minimizes the sums of distances from the hyperplane and not the number of errors (as it corresponds to an NP-complete problem)

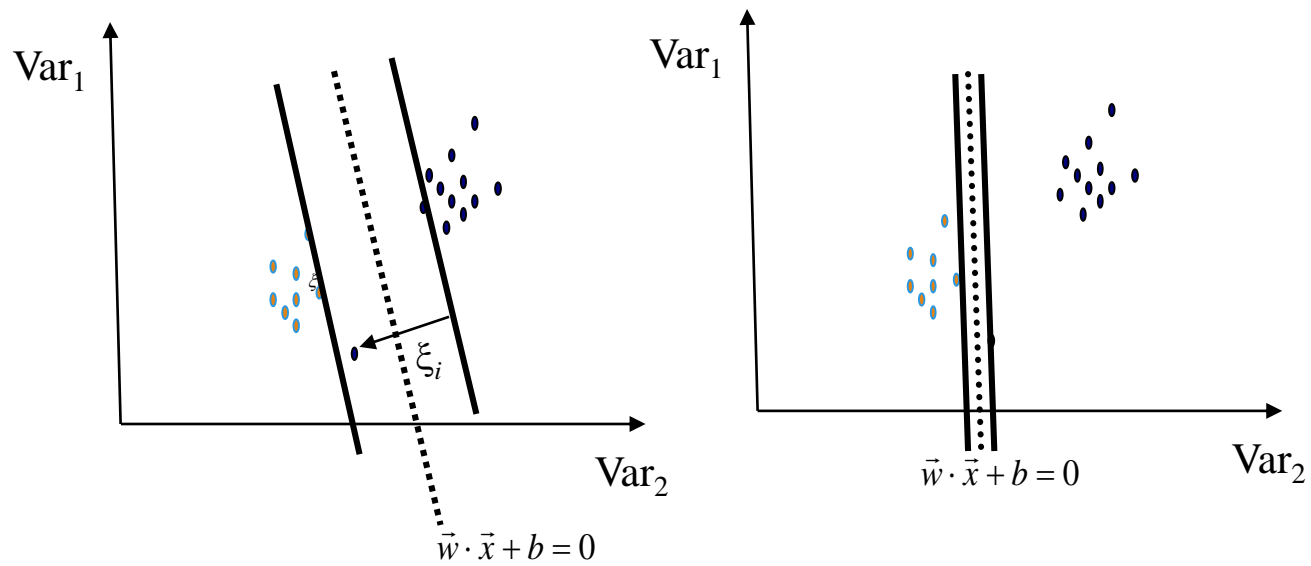
If $C \rightarrow \infty$, the solution tends to conform to the hard margin solution

ATT!!!!: if $C = 0$ then $\|\vec{w}\| = 0$. Infact it is always possible to satisfy:

$$y_i b \geq 1 - \xi_i \quad \forall \vec{x}_i$$

If C grows, it tends to limit the number of tolerated errors. Infinite settings for C provide the number of errors to be 0, exactly as in the *hard-margin* formulation.

Robustness: *Soft vs Hard Margin SVMs*



Soft Margin SVM

Hard Margin SVM

Soft vs Hard Margin SVMs

A Soft-Margin SVM has always a solution

A Soft-Margin SVM is more robust wrt odd training examples

- *Insufficient Representation (e.g. Limited Vocabularies)*
- *High ambiguity of (linguistic) features*

An Hard-Margin SVM requires no parameter

References

Basili, R., A. Moschitti Automatic Text Categorization: From Information Retrieval to Support Vector Learning , Aracne Editrice, Informatica, ISBN: 88-548-0292-1, 2005

A tutorial on Support Vector Machines for Pattern Recognition (C.J.Burges)

- URL: <http://www.umiacs.umd.edu/~joseph/support-vector-machines4.pdf>

The Vapnik-Chervonenkis Dimension and the Learning Capability of Neural Nets (E.D: Sontag)

- URL: http://www.math.rutgers.edu/~sontag/FTP_DIR/vc-expo.pdf

Computational Learning Theory

- (Sally A Goldman Washington University St. Louis Missouri)
- <http://www.learningtheory.org/articles/COLTSurveyArticle.ps>

AN INTRODUCTION TO SUPPORT VECTOR MACHINES (and other kernel-based learning methods),
N. Cristianini and J. Shawe-Taylor Cambridge University Press.

The Nature of Statistical Learning Theory, V. N. Vapnik - Springer Verlag (December, 1999)